An Approach to Making SPAI and PSAI Preconditioning Effective for Large Irregular Sparse Linear Systems*

Zhongxiao Jia† Qian Zhang‡

Abstract

We investigate the SPAI and PSAI preconditioning procedures and shed light on two important features of them: (i) For the large linear system $Ax = b$ with A irregular sparse, i.e., with A having s relatively dense columns, SPAI may be very costly to implement and the resulting sparse approximate inverses may be ineffective for preconditioning. PSAI can be effective for preconditioning but may require excessive storage and be unacceptably time consuming; (ii) the situation is improved drastically when A is regular sparse, that is, all of its columns are sparse. In this case, both SPAI and PSAI are efficient and more likely to construct effective preconditioners. Motivated by these features, we propose an approach to making SPAI and PSAI more practical for $Ax = b$ with A irregular sparse. We first split A into a regular sparse $\tilde{A}$ and a matrix of low rank s. Then exploiting the Sherman–Morrison–Woodbury formula, we transform $Ax = b$ into $s+1$ new linear systems with the same coefficient matrix $\tilde{A}$, use SPAI and PSAI to compute sparse approximate inverses of $\tilde{A}$ efficiently and apply Krylov iterative methods to solve the preconditioned linear systems. We show how to recover an approximate solution of $Ax = b$ from those of the $s+1$ new systems. Based on the relationship between approximate solutions, we design reliable stopping criteria for the $s+1$ systems to guarantee that the approximate solution of the original $Ax = b$ satisfies a desired accuracy. Given the fact that irregular sparse linear systems are common in applications, this approach widely extends the practicability of SPAI and PSAI. Numerical results demonstrate the considerable superiority of our approach to the direct application of SPAI and PSAI to $Ax = b$.

Key words. Preconditioning; sparse approximate inverse; irregular sparse; regular sparse; the Sherman–Morrison–Woodbury formula; F-norm minimization; Krylov solver

AMS Subject Classification (2000). 65F10

1 Introduction

Krylov iterative solvers [9, 17] have been very popular for the large sparse linear system

$$Ax = b, \tag{1}$$

where A is a nonsingular $n \times n$ matrix and b is an n-dimensional vector. However, when A has bad spectral property or is ill conditioned, the solvers generally exhibit extremely slow convergence and necessitate preconditioning techniques. Sparse approximate inverse (SAI) preconditioning is for general purpose and aims to compute a preconditioner $M \approx A^{-1}$

*Supported by National Basic Research Program of China 2011CB302400 and the National Science Foundation of China (No. 11071140).

†Department of Mathematical Sciences, Tsinghua University, Beijing 100084, People's Republic of China, jiazx@tsinghua.edu.cn.

‡Department of Mathematical Sciences, Tsinghua University, Beijing 100084, People's Republic of China, qianzhang.thu@gmail.com.

directly so as to improve the conditioning of (1) for the vast majority of problems [3, 17]. One typical SAI preconditioning technique is the adaptive sparse approximate inverse (SPAI) procedure [11], which has been widely used and proven quite effective for a broad range of matrices. The recently proposed adaptive power sparse approximate inverse (PSAI) procedure with dropping [15], called PSAI(tol), is also an effective SAI preconditioning technique and has been shown to be at least competitive with SPAI numerically and can outperform SPAI for some practical problems.

The SPAI and PSAI(tol) procedures fall into the category of Frobenius norm (hereafter abbreviated as F-norm) minimization based SAI preconditioning. During loops they solve a sequence of constrained optimization problems of the form

$$\min_{M\in\mathcal{M}} \|AM - I\|_F, \tag{2}$$

where $\mathcal{M}$ is the set of matrices with a given sparsity pattern $\mathcal{S}$. Denote by $\mathcal{M}_k$ the set of n-dimensional vectors whose sparsity pattern is $\mathcal{S}_k = \{i|(i,k) \in \mathcal{S}\}$. Then (2) is decoupled into n independent constrained least squares (LS) problems

$$\min_{m_k\in\mathcal{M}_k} \|Am_k - e_k\|, \quad k = 1, 2, \ldots, n, \tag{3}$$

with e_k the k-th column of the $n \times n$ identity matrix I. Here and hereafter, the norm $\|\cdot\|$ denotes the vector 2-norm or the matrix spectral norm. For each k, let $\tilde{m}_k = m_k(\mathcal{S}_k)$, $\mathcal{L}_k$ be the set of indices of nonzero rows of $A(:, \mathcal{S}_k)$, $\tilde{A}_k = A(\mathcal{L}_k, \mathcal{S}_k)$ and $\tilde{e}_k = e_k(\mathcal{L}_k)$. Then (3) is reduced to the smaller unconstrained LS problems

$$\min_{\tilde{m}_k} \|\tilde{A}_k\tilde{m}_k - \tilde{e}_k\|, \quad k = 1, 2, \ldots, n, \tag{4}$$

which can be solved by the QR decomposition in parallel. SPAI and PSAI(tol) determine the sparsity pattern $\mathcal{S}_k$ dynamically: starting with a simple initial pattern, which is usually taken as the pattern of e_k, $\mathcal{S}_k$ is augmented or adjusted adaptively until the residual norm $\|Am_k - e_k\|$ falls below a given tolerance or the maximum number of augmentations is reached. The distinction between SPAI and PSAI(tol) lies in the way that $\mathcal{S}_k$ is augmented or adjusted. The already existing positions of nonzero entries in m_k for SPAI are retained in the later loops and only a few new members, called most profitable indices in [11], are added per loop. PSAI(tol) aims to drop the small entries whose sizes are below a certain tolerance and only retain the remaining large ones during the determination of M, and $\mathcal{S}_k$ is adjusted dynamically by not only absorbing new members but also discarding the positions where the entries of m_k become small during loops. Because of these essential differences, PSAI(tol) may better capture effective sparsity patterns of A^{-1} than SPAI.

Throughout the paper, we will frequently use two keywords "regular sparse" and "irregular sparse" for a matrix. By regular sparse, we, qualitatively and sensibly, mean that all the columns of the matrix are sparse and no column has much more nonzero entries than the others. By irregular sparse, we mean that there are some (relatively) dense columns, each of which has considerably more nonzero entries than the other sparse columns. We call such columns irregular and denote by s the number of them. Problem (1) with A irregular sparse are quite common and arises in semiconductor device problem, power network problem, circuit simulation, optimization problems, computer graphics/vision problem, economic problems, material problems, theoretical/quantum chemistry problem and many others [7]. Quantitatively, we will declare a matrix irregular sparse if its densest column has $10p$ entries or more, where p is the average number of nonzero entries each column. We will defer quantitative details on these two keywords to Section 3. Under this definition, we investigate all

the matrices in the collection [7], which contains 2649 matrices and of them 1978 are square. We find that 682 out of these 1978 matrices are irregular sparse. That is, there are 34% of the matrices in the collection which are irregular sparse. In the collection, there are some social networks, citation networks, and other graphs that are not typically viewed as linear systems. They often have dense columns. But even if these matrices are removed, there are 30% irregular matrices, 464 out of 1554, in the collection. So irregular sparse linear systems are quite common in applications. We will give more illustrative details on irregular matrices in Section 4, where we clearly show where irregular sparse linear systems are from, how big s can be and how dense irregular columns.[1]

The possible success of any SAI preconditioning procedure is based on the crucial assumption that A has good sparse approximate inverses. Under this assumption throughout the paper, we consider the case that A is irregular. It is empirically observed that good sparse approximate inverses of A are irregular too. Nevertheless, since the number of nonzero entries in any single column of the final M in SPAI is bounded by the number of most profitable indices per loop times the maximum loop steps, a column of M may have not enough or excessive nonzero entries. As a result, M's obtained by SPAI may not approximate A^{-1} well and thus may be ineffective for preconditioning, or it is a good sparse approximate inverse of A but some columns of it are denser than they should be. We will justify theoretically in subsection 2.1 and confirm numerically in subsection 4.1 that the SPAI algorithm may be very costly to implement. For PSAI(tol), we may also suffer the unaffordable overhead from solving some possibly large LS problems (4), although it is more likely to construct effective preconditioners no matter wether A is regular sparse or not. Remarkably, it turns out that the situation mentioned above is improved very substantially if A is regular sparse. With reasonable parameters, it is not only possible to implement SPAI and PSAI(tol) efficiently but also to obtain possibly effective sparse approximate inverses. In Section 2, we will give detailed arguments to clarify these claims.

It is well known [5] that the computational consumption, stability and effectiveness of factorized SAI preconditioners are generally sensitive to reorderings of A. Unfortunately, it appears that reorderings do not help for SPAI and PSAI(tol). The reason is that reorderings do not change the irregularity of sparsity patterns of A, A^{-1} and good sparse approximate inverses of A. Therefore, with orderings, SPAI and PSAI(tol) may still be very costly to implement, and SPAI may still be ineffective for preconditioning. We refer the reader to [3] for the relevant arguments about SPAI, which are valid for PSAI(tol) as well.

Based on our previous statements, we naturally come up with the idea of transforming the irregular sparse problem (1) into some regular sparse one(s), on which SPAI and PSAI(tol) work well. It will appear that the Sherman–Morrison–Woodbury formula [10, 18] is a key step towards our goal and provides us a powerful tool. We present an approach to splitting A into a regular sparse matrix $\tilde{A}$ and a matrix of low rank s and transforming (1) into the $s+1$ new linear systems with the same coefficient matrix $\tilde{A}$. By exploiting the Sherman–Morrison–Woodbury formula, we can recover the solution of (1) from the ones of the $s+1$ systems directly. We consider numerous practical issues on how to obtain a desired splitting of A, how to define and compute an approximate solution of (1) via those of the new systems and how accurately we should solve the new systems. A remarkable merit of this approach is that SPAI and PSAI(tol) can construct effective preconditioners for the new systems efficiently, making Krylov solvers converge fast. The price we pay is to solve $s+1$ linear systems. But a great bonus is that we only need to construct one effective sparse approximate inverse M efficiently for the $s+1$ systems. The price is generally insignificant as it is typical that

[1] We thank Professor Davis, one of the authors of [7] very much for providing us such a very valuable data analysis, clearly showing that irregular sparse linear problems are quite common.

the construction of M dominates the whole cost of preconditioned Krylow solvers even in a parallel computing environment. As a matter of fact, due to inherent parallelizations of SPAI and PSAI(tol), SAI type precondtioners are attractive for solving a sequence of linear systems with the same coefficient matrix, as has been addressed in the literature, e.g., [4]. As a consequence, given the fact that irregular sparse linear systems are quite common in applications, our approach widely extends the practicality of SPAI and PSAI.

The paper is organized as follows. In Section 2, we review SPAI and PSAI(tol) procedures and shed light on the above-mentioned features for A irregular sparse and regular sparse, respectively. In Section 3, we describe our new approach for solving (1) with A irregular sparse and address some theoretical and practical considerations. In Section 4, we report numerical experiments to demonstrate the superiority of our approach to SPAI and PSAI(tol) applied to (1) directly and the superiority of PSAI(tol) to SPAI for both irregular and regular sparse linear systems. Finally, we conclude the paper in Section 5.

2 The SPAI and PSAI(tol) procedures

In this section, we give an overview of the SPAI and PSAI(tol) procedures and shed light on the facts that SPAI and PSAI(tol) are costly, may be impractical to implement and SPAI may not be effective for preconditioning if A is irregular sparse.

2.1 The SPAI procedure

Denote by $\mathcal{S}_k^{(l)}$ the sparsity pattern of m_k after l loops of augmentation starting with a given initial pattern $\mathcal{S}_k^{(0)}$, and by $\mathcal{L}_k^{(l)}$ the set of indices of nonzero rows of $A(:, \mathcal{S}_k^{(l)})$. Let $\tilde{A}_k = A(\mathcal{L}_k^{(l)}, \mathcal{S}_k^{(l)}), \tilde{e}_k = e_k(\mathcal{L}_k^{(l)})$, $\tilde{m}_k$ be the solution of (4). Then the residual of (3) is

$$r = A(:, \mathcal{S}_k^{(l)})\tilde{m}_k - e_k.$$

For $r \neq 0$, denote by $\mathcal{L}$ the set of indices l for which $r(l) \neq 0$, and by $\mathcal{N}$ the set of indices of nonzero columns of $A(\mathcal{L}, :)$. Then

$$\tilde{\mathcal{J}} = \mathcal{N} \backslash \mathcal{S}_k^{(l)} \tag{5}$$

constitutes the new candidates for augmenting $\mathcal{S}_k^{(l)}$ in the next loop. Grote and Huckle [11] suggest to select several most profitable indices from $\tilde{\mathcal{J}}$ and augment them to $\mathcal{S}_k^{(l)}$ to obtain a new sparsity pattern $\mathcal{S}_k^{(l+1)}$ of m_k. They do this as follows: for each $j \in \tilde{\mathcal{J}}$, consider the one-dimensional minimization problem

$$\min_{\mu_j} \|r + \mu_j A e_j\|, \tag{6}$$

whose solution is

$$\mu_j = -\frac{r^T A e_j}{\|A e_j\|^2}, \tag{7}$$

where the superscript T denotes the transpose of a vector or matrix. The 2-norm ρ_j of the new residual $r + \mu_j A e_j$ satisfies

$$\rho_j^2 = \|r\|^2 - \frac{(r^T A e_j)^2}{\|A e_j\|^2.} \tag{8}$$

The set $\tilde{\mathcal{S}}_k^{(l)}$ of the most profitable indices j consists of those associated with a few, say, 1 to 5, smallest ρ_j's and is added to $\mathcal{S}_k^{(l)}$ to obtain $\mathcal{S}_k^{(l+1)}$. Update $\mathcal{L}_k^{(l)}$ to get $\mathcal{L}_k^{(l+1)}$ by adding

the set $\tilde{\mathcal{L}}_k^{(l)}$ of indices of new nonzero rows corresponding to $\tilde{S}_k^{(l)}$. The new augmented LS problem (4) is solved by updating $\tilde{m}_k$ instead of resolving it. Proceed in such a way until $\|Am_k - e_k\| < \delta$ or l attains the prescribed maximum loop steps $l_{\max}$, where δ is a given fairly small tolerance, say $0.1 \sim 0.4$.

For a given small maximum loop steps $l_{\max}$, Huckle [13] has established effective upper bounds for the pattern of M obtained by SPAI, and shown that the patterns of $(A^TA)^{l_{\max}}A^T$, $(I + |A| + |A^T|)^{l_{\max}}A^T$ and $(I + A)^{l_{\max}}$ are good envelop patterns of M.

Remark 1. Let us consider the computational complexity of SPAI. When A is regular sparse, it is straightforward to verify that $r = Am_k - e_k$ is also sparse and both $\mathcal{L}$ and $\tilde{\mathcal{J}}$ have only a few elements. As a consequence, the cardinal numbers of $\mathcal{S}_k^{(l)}$'s are small, so are the orders of $\tilde{A}_k$'s in (4). Therefore, it is cheap to determine the set $\tilde{\mathcal{S}}_k^{(l)}$ of the most profitable indices and solve (4). However, the situation deteriorates severely when A is irregular sparse. For example, assume that the k-th column a_k of A is relatively dense, and suppose $A(k,k) \neq 0$ and take $\mathcal{S}_k^{(0)} = \{k\}$. Then in the first loop of SPAI, $r = m_k(k)a_k - e_k$ is as dense as a_k, causing that $\mathcal{L}$ has a very big cardinal number and $\tilde{\mathcal{J}}$ has a lot of elements. This situation is extended in subsequent loops. As a consequence, suppose that a_k is fully dense, at each loop we have to compute almost n numbers ρ_j's, order them and select the most profitable indices in $\tilde{\mathcal{J}}$. So SPAI may be very expensive whenever A is irregular sparse.

Remark 2. For the irregular sparse A, suppose that it has good sparse approximate inverses. Then they are typically irregular sparse too. Suppose the k-th column of a good sparse approximate inverse is irregular. The description of SPAI shows that if a_k is irregular, i.e., relatively dense, then the set $\tilde{\mathcal{J}}$ in (5) has big cardinal number during loops. Nevertheless, SPAI simply takes the set $\tilde{\mathcal{S}}_k^{(l)}$ to be *only a few* most profitable indices from $\tilde{\mathcal{J}}$. If the cardinality of $\tilde{\mathcal{S}}_k^{(l)}$ is fixed small, say 5, the default value as suggested and used in [1, 2, 11], then the k-th column of M is sparse and may not approximate the k-th column of A^{-1} well unless the loop steps $l_{\max}$ is large enough. Since the number of nonzero entries in any single column of the final M in SPAI is bounded by the number of most profitable indices per loop times the maximum loop steps $l_{\max}$, which is fairly small, say 20, the default used in [1, 2], a column of M may have not enough or excessive nonzero entries. As a result, M's obtained by SPAI may not approximate A^{-1} well and thus may be ineffective for preconditioning, or it is a good sparse approximate inverse of A but some columns of it are denser than they should be. On the other hand, if the cardinality of $\tilde{\mathcal{S}}_k^{(l)}$ is allowed to vary and instead taken big, the resulting M may be a good preconditioner. However, we have to solve some LS problems (4) of large sizes, causing that SPAI may be very costly and even impractical to implement.

2.2 The PSAI(tol) procedure

PSAI(tol) is different from SPAI in the way that selects and adjusts $\mathcal{S}$, i.e., $\mathcal{S}_k$, $k = 1, 2, \ldots, n$. We first review the basic PSAI (BPSAI) procedure [15]. From the Cayley–Hamilton theorem, A^{-1} can be expressed as a matrix polynomial of A of degree $d - 1$ with $d \leq n$:

$$A^{-1} = \sum_{i=0}^{d-1} c_i A^i \tag{9}$$

with $A^0 = I$ and c_i $(i = 0, 1, \ldots, d-1)$ being certain constants. Denote $\mathcal{P}(\cdot)$ by the sparsity pattern of a matrix or vector. It is obvious that $\mathcal{P}(A^{-1}) \subseteq \mathcal{P}((I + A)^{d-1})$. For a given small positive number $l_{\max} \ll d$, the pattern $\mathcal{S}$ of the sparse approximate inverse M is taken as a subset of $\mathcal{P}((I + A)^{l_{\max}})$ in BPSAI. Therefore, the pattern $\mathcal{S}_k$ of the k-th column m_k of M,

is a subset of $\bigcup_{l=0}^{l_{\max}} \mathcal{P}(A^l e_k)$ since

$$\mathcal{P}((I+A)^{l_{\max}}) = \bigcup_{l=0}^{l_{\max}} \mathcal{P}(A^l).$$

The adjustment of $\mathcal{S}_k$ and the construction of m_k for $1 \le k \le n$ proceed dynamically as follows. For $l = 0, 1, \ldots, l_{\max}$, denote by $\mathcal{S}_k^{(l)}$ the sparsity pattern of m_k at the current loop step l and by $\mathcal{L}_k^{(l)}$ the set of indices of nonzero rows of $A(:, \mathcal{S}_k^{(l)})$. Set $a_k^l = A^l e_k$. Then $a_k^{l+1} = A a_k^l$. So the pattern of a_k^{l+1} can be determined. The sparsity pattern $\mathcal{S}_k^{(l+1)}$ is updated as $\mathcal{S}_k^{(l+1)} = \mathcal{P}(a_k^{l+1}) \cup \mathcal{S}_k^{(l)}$. In the next loop we solve the augmented LS problem

$$\min \|A(\mathcal{L}_k^{(l+1)}, \mathcal{S}_k^{(l+1)}) m_k(\mathcal{S}_k^{l+1}) - e_k(\mathcal{S}_k^{l+1})\|$$

with $\mathcal{L}_k^{(l+1)} = \mathcal{L}_k^{(l)} \cup \tilde{\mathcal{L}}_k^{(l)}$, where $\tilde{\mathcal{L}}_k^{(l)}$ is the set of indices of new nonzero rows corresponding to the set $\mathcal{P}(a_k^{l+1}) \backslash \mathcal{S}_k^{(l)}$ of indices of the newly added columns. This problem corresponds to the small LS problem (4), whose solution can be updated from $m_k(\mathcal{S}_k^l)$ efficiently. Proceed in such a way until $\|Am_k - e_k\| < \delta$ or $l > l_{\max}$.

It has been proved in [15, Theorem 1] that for $\mathcal{S}_k^{(0)} = \{k\}$, as $l_{\max}$ increases, M obtained by BPSAI may become denser and denser quickly once one column in A is irregular sparse. In order to control the sparsity of M and construct an effective preconditioner, some reasonable dropping strategies should be used. Two practical PSAI(*lfill*) and PSAI(*tol*) algorithms have been proposed in [15]. PSAI(*lfill*) aims to retain at most *lfill* large entries of m_k. To be more flexible, *lfill* may vary with k. Since the numbers of large entries in the columns of A^{-1} are generally unknown in advance, it is difficult to choose a reasonable *lfill* in advance. This is a shortcoming similar to SPAI. In contrast, for the newly computed m_k at loop l, PSAI(*tol*) drops those small entries below a prescribed tolerance *tol* and retains only large ones. Therefore, PSAI(*tol*) is more reasonable and reliable to capture an effective approximate sparsity pattern of A^{-1} and determine its large entries. A central issue is the selection of dropping tolerance *tol*. This issue is mathematically nontrivial, and *tol* has strong effects on the effectiveness of PSAI(*tol*) and many other SAI preconditioning procedures. Very recently, the authors [14] have established a systematic and solid theory on dropping criteria for PSAI(*tol*) and all the static F-norm minimization based SAI preconditioning procedures. For PSAI(*tol*), the authors show that, at step $l \le l_{\max}$, a nonzero entry m_{jk} is dropped for $1 \le j \le n$ if it satisfies

$$|m_{jk}| \le tol_k = \frac{\delta}{nnz(m_k)\|A\|_1}, \ k = 1, 2, \ldots, n, \tag{10}$$

where $nnz(\cdot)$ is the number of nonzero entries in a vector or matrix and δ is the stopping tolerance for SPAI and BPSAI. We remind that m_k in (10) is the newly computed one at loop l. This criterion has been shown to be robust and effective, making M as sparse as possible and have similar preconditioning quality to the possibly much denser one obtained by BPSAI. More precisely, for the final M obtained by PSAI(*tol*), it is proved in [14] that if (10) is used then the residual norm $\|Am_k - e_k\| < 2\delta, \ k = 1, 2, \ldots, n$ under the assumption that the the residual norm of each column of the preconditioner obtained by BPSAI falls below δ.

The dropping criterion (10) also applies to any static F-norm minimization based SAI preconditioning procedure [14], where δ is replaced by $\delta_k = \|Am_k - e_k\|$ and m_k is computed

by the static SAI preconditioning procedure without dropping for an a-priori given sparsity pattern $\mathcal{S}_k$. Write the sparsification of m_k as m_k in such a way. Then it is known [14] that $\|Am_k - e_k\| \leq 2\delta_k,\ k = 1, 2, \ldots, n$.

Remark. If A is regular sparse, it is direct to justify that the size of $\tilde{A}_k$ is small for a small $l_{\max}$ and the cost for solving (4) is cheap. However, the situation is sharply different when A is irregular sparse. Suppose that the k-th column a_k of A is relatively dense and $A(k,k) \neq 0$, and let $\mathcal{S}_k^{(0)} = \{k\}$. Then the corresponding m_k is also relatively dense since its pattern $\mathcal{S}_k^{(1)} = \mathcal{P}(a_k) \cup \{k\} = \mathcal{P}(a_k)$ at the first loop. Therefore, (4) is a relatively large LS problem no matter whether dropping is used or not. Generally, if the number of nonzero entries in a_k is comparable to n, then we have to solve a LS problem whose size is comparable to n, causing that PSAI(tol) is impractical.

3 Transformation of (1) into regular sparse linear systems

As we have already seen, when A is irregular sparse, both the SPAI and PSAI(tol) procedures encounter some serious difficulties and M obtained by SPAI may be ineffective for preconditioning (1). In this section, we will attempt to transform (1) into some regular sparse ones, for which it is possible to use SPAI and PSAI(tol) to construct effective preconditioners efficiently. It turns out that the Sherman–Morrison–Woodbury formula will play a central role for our purpose. We will address numerous practical issues on the determination of irregular columns and the construction of $\tilde{A}$, etc.

We now review the Sherman–Morrison–Woodbury formula [18, p.330].

Theorem 1. *Let $U, V \in \mathbb{R}^{n\times s}$ with $s \leq n$. If A and $A - UV^T$ are nonsingular, then $I - V^TA^{-1}U$ is nonsingular and*

$$(A - UV^T)^{-1} = A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}. \tag{11}$$

In particular, if $U = u$ and $V = v$ are vectors, then

$$(A - uv^T)^{-1} = A^{-1} + \frac{A^{-1}uv^TA^{-1}}{1 - v^TA^{-1}u}. \tag{12}$$

If $s < n$ and U and V are of full column rank, the theorem says that a rank s correction to A results in a rank s correction to A^{-1}. We should comment that the theorem itself requires neither of the two conditions. However, the formula is typically of interest for $s \ll n$, and with $s = 1$ it reduces to the Sherman–Morrison formula. For a good survey on the history and some applications, we refer the reader to [12].

For our purpose, assume that the $j_1, j_2, \ldots, j_s$-th columns of A are irregular and the remaining $n - s$ ones are sparse. Denote by $A_{dc} = (a_{j_1}, a_{j_2}, \ldots, a_{j_s})$ the matrix consisting of the s irregular columns of A, by $\tilde{A}_{dc} = (\tilde{a}_{j_1}, \tilde{a}_{j_2}, \ldots, \tilde{a}_{j_s})$ the sparsification of A_{dc} that drops some of its nonzero entries, so that each column of $\tilde{A}_{dc}$ is as sparse as the other $n-s$ columns of A. Define $U = A_{dc} - \tilde{A}_{dc} = (u_1, u_2, \ldots, u_2)$. Then U just consists of those dropped nonzero entries of A_{dc}. Denote by $\tilde{A}$ the regular sparse matrix that replaces a_{j_i} by $\tilde{a}_{j_i}$, $i = 1, 2, \ldots, s$. Then A is split into

$$A = \tilde{A} + UV^T, \tag{13}$$

where $V = (e_{j_1}, e_{j_2}, \ldots, e_{j_s})$ and e_{j_i} is the the j_i-th column of the $n \times n$ identity matrix I, $i = 1, 2, \ldots, s$. Assume that $\tilde{A}$ is nonsingular. Then it follows from Theorem 1 that $I + V^T\tilde{A}^{-1}U$ is nonsingular and

$$A^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1}U(I + V^T\tilde{A}^{-1}U)^{-1}V^T\tilde{A}^{-1}. \tag{14}$$

Therefore, the solution of (1) is

$$x = A^{-1}b = \tilde{A}^{-1}b - (\tilde{A}^{-1}U)(I + V^T(\tilde{A}^{-1}U))^{-1}(V^T\tilde{A}^{-1}b). \tag{15}$$

This amounts to solving a new regular sparse linear system

$$\tilde{A}y = b \tag{16}$$

and the other s regular sparse linear systems

$$\tilde{A}w_j = u_j,\ j = 1, 2, \ldots, s. \tag{17}$$

If the exact solutions to (16) and (17) were available, we would compute the solution x of (1) from (15) by solving the small $s \times s$ linear system with the coefficient matrix $I + V^T(\tilde{A}^{-1}U)$ and the right-hand side $V^T\tilde{A}^{-1}b$. This could be done by any numerically stable direct solver very cheaply.

We can summarize the above approach as Procedure *.

Procedure *: Solving the irregular sparse linear system (1)

1: Find s and A_{dc} and sparsify A_{dc} to get $\tilde{A}_{dc}$. Define $U = A_{dc} - \tilde{A}_{dc}$ and the regular sparse matrix $\tilde{A} = A - UV^T$, where $V = (e_{j_1}, e_{j_2}, \ldots, e_{j_s})$. Assume that $\tilde{A}$ is nonsingular.
2: Solve $s + 1$ linear systems (16) and (17) for y and $w_1, w_2, \ldots, w_s$, respectively.
3: Let $W = (w_1, w_2, \ldots, w_s)$ and compute the solution x of $Ax = b$ by

$$x = A^{-1}b = y - W(I + V^TW)^{-1}(V^Ty). \tag{18}$$

Remark. If A has only one irregular column and $j_1 = k$, we only need to solve two linear systems $\tilde{A}y = b$ and $\tilde{A}w = u$, and (18) reduces to $x = A^{-1}b = y - \frac{e_k^T y}{1+e_k^T w}w$.

For regular sparse systems (16) and (17), we suppose that only iterative solvers are viable in our context. Now, a big and direct reward is that SPAI and PSAI(tol) can be used much more efficiently to construct effective preconditioners for the $s+1$ regular sparse systems than the original irregular sparse (1). So it is expected that the preconditioned Krylov solvers can converge quickly.

In order to use Procedure * to develop a practical iterative solver for (1), we need to handle several practical issues. Also, we will see that recovering an *approximate* solution of (1) via those of (16) and (17) is involved and is not as simple as Procedure * indicates, in which the exact y and W are assumed.

The first issue is about the quantitative meaning of irregular columns, by which we define A_{dc} and U. Obviously, like sparsity itself and many other quantities in numerical analysis, it appears impossible to give a precise definition of it. In fact, it is also unnecessary to do so. In our experiments, we empirically find that the threshold $10p$ is a good choice, where p is the average number of nonzero entries per column of A. If the number of nonzero entries in a column exceeds $10p$, then we regard the column as an irregular one. Based on this criterion, we determine all the irregular columns of A and the number s of them. We point out that other thresholds ranging from $8p$ to $15p$ work equally well. So the efficiency and effectiveness of SPAI and PSAI(tol) is quite insensitive to thresholds. This is very appealing for the robustness and generality of our algorithm.

The second issue is which nonzero entries in A_{dc} should be dropped to get $\tilde{A}_{dc}$ and generate a regular matrix $\tilde{A}$. In principle, the number $\hat{p}$ of nonzero entries in each column

of $\tilde{A}_{dc}$ should be comparable to the average number p of nonzero entries per column of A. Given $\hat{p}$, there may be many dropping ways. We propose two approaches. The first one is to retain the diagonal and $\hat{p}-1$ nonzero entries nearest to the diagonal and drop the others in each column of A_{dc}. The second approach is to retain the diagonal and the other $\hat{p}-1$ largest entries and drop the others in each column of A_{dc}. For the choice of $\hat{p}$, to be unique, we suppose taking $\hat{p}=p$ simply. Numerically, two approaches have exhibited very similar behavior. We will only report the results obtained by the first approach.

The third issue is about the sparsity, non-singularity and conditioning of $\tilde{A}$ and the existence of good sparse approximate inverses of $\tilde{A}$. As is clear from its construction, $\tilde{A}$ is sparser than A. For the nonsingular $\tilde{A}$, although we cannot theoretically guarantee its non-singularity, $\tilde{A}$ should be nonsingular and the (numerically) singular case must be rare in practice. As for the conditioning of $\tilde{A}$, it is possible to get either a well-conditioned or ill-conditioned $\tilde{A}$. Given that A is ill conditioned, the former is naturally welcome, but we should not expect such a luck; the latter is more possible. Theoretically speaking, $\tilde{A}$ may be worse or better conditioned than A. Happily, numerical experiments will indicate that $\tilde{A}$'s are often, though not always, better conditioned than A's. Actually, what we have found is that for 8 out of 12 test matrices, $\tilde{A}$'s are considerably better conditioned than A's and for the other four matrices, the condition numbers of each $\tilde{A}$ and the corresponding A are very near. Regarding the existence of good sparse approximate inverses of $\tilde{A}$, we, informally and heuristically, argue as follows: for a given positive integer $l_{\max}$, the pattern $\mathcal{P}((I+\tilde{A})^{l_{\max}})$ is generally sparser than $\mathcal{P}((I+A)^{l_{\max}})$. Since the sparsity patterns of approximate inverses obtained by SPAI and PSAI(tol) are effectively bounded by $\mathcal{P}((I+\tilde{A})^{l_{\max}})$ and $\mathcal{P}((I+A)^{l_{\max}})$, respectively, it is expected that $\tilde{A}$ has good sparse approximate inverses when A does.

The last important issue is how we should select stopping criteria for Krylov iterations for $s+1$ linear systems so as to recover an approximate solution of (1) with the prescribed accuracy. It is seen from (18) that the solution x of (1) is formed from the ones of the $s+1$ new systems. Recall that the $s+1$ linear systems are now supposed to be solved approximately by preconditioned Krylov solvers. Our concerns are (i) how to define an approximate solution $\tilde{x}$ from the $s+1$ approximate solutions of (16) and (17) and (ii) how accurately we should solve (16) and (17) such that $\tilde{x}$ satisfies $\frac{\|r\|}{\|b\|}=\frac{\|b-A\tilde{x}\|}{\|b\|}<\varepsilon$. As it appears immediately, it is direct to settle down the first concern, but the second concern is involved.

Theorem 2. *Let $\tilde{y}$ and $\tilde{w}_j,\ j=1,\ldots,s$ be the approximate solutions of $\tilde{A}y=b$ and $\tilde{A}w_j=u_j,\ j=1,2,\ldots,s$, respectively, and define $\tilde{W}=(\tilde{w}_1,\ldots,\tilde{w}_s)$, $r_{\tilde{y}}=b-\tilde{A}\tilde{y}$, $r_{\tilde{w}_j}=u_j-\tilde{A}\tilde{w}_j$. Assume that $I+V^T\tilde{W}$ is nonsingular with $V=(e_{j_1},e_{j_2},\ldots,e_{j_s})$, and define $c=\|(I+V^T\tilde{W})^{-1}(V^T\tilde{y})\|$. Let*

$$\tilde{x}=\tilde{y}-\tilde{W}(I+V^T\tilde{W})^{-1}(V^T\tilde{y}) \tag{19}$$

be an approximate solution of (1). *Then if*

$$\frac{\|r_{\tilde{y}}\|}{\|b\|}<\frac{\varepsilon}{2} \tag{20}$$

and

$$\frac{\|r_{\tilde{w}_j}\|}{\|u_j\|}<\frac{\|b\|}{2\sqrt{s}c\|u_j\|}\varepsilon,\ j=1,2,\ldots,s, \tag{21}$$

we have

$$\frac{\|r\|}{\|b\|}=\frac{\|b-A\tilde{x}\|}{\|b\|}<\varepsilon. \tag{22}$$

Proof. Replacing W and y by their approximations $\tilde{W}$ and $\tilde{y}$ in (18), we get (19), which is naturally an approximate solution of (1). Define $R_{\tilde{W}} = U - \tilde{A}\tilde{W}$. We obtain

$$\begin{aligned}
r &= b - A\tilde{x} = b - A\tilde{y} + A\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= b - (\tilde{A} + UV^T)\tilde{y} + (\tilde{A} + UV^T)\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= r_{\tilde{y}} - UV^T\tilde{y} + UV^T\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) + \tilde{A}\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= r_{\tilde{y}} - U(I - V^T\tilde{W}(I + V^T\tilde{W})^{-1})(V^T\tilde{y}) + \tilde{A}\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= r_{\tilde{y}} - U(I - (I + V^T\tilde{W} - I)(I + V^T\tilde{W})^{-1})(V^T\tilde{y}) + \tilde{A}\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= r_{\tilde{y}} - U(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) + \tilde{A}\tilde{W}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}) \\
&= r_{\tilde{y}} - R_{\tilde{W}}(I + V^T\tilde{W})^{-1}(V^T\tilde{y}),
\end{aligned}$$

from which it follows that

$$\|r\| \le \|r_{\tilde{y}}\| + c\|R_{\tilde{W}}\| \le \|r_{\tilde{y}}\| + c\|R_{\tilde{W}}\|_F. \tag{23}$$

By definition and $R_{\tilde{W}}(:, j) = r_{\tilde{w}_j}$, we have $\|R_{\tilde{W}}\|_F = \sqrt{\sum_{j=1}^{s} \|r_{\tilde{w}_j}\|^2}$. If (20) and

$$\frac{\|r_{\tilde{w}_j}\|}{\|b\|} < \frac{\varepsilon}{2\sqrt{s}c}, \ j = 1, 2, \ldots, s$$

(23) means that (22) holds. Since the above relation is just (21), the theorem holds. □

We point out that because of (23) our stopping criterion (21) may be conservative. Furthermore, we comment that c is moderate if $I + V^T\tilde{W}$ is well conditioned and it may be large if $I + V^T\tilde{W}$ is ill conditioned. Since s is supposed very small, this theorem indicates that we should solve the s linear systems (17) roughly with the accuracy at the level of ε. We may solve them by Krylov solvers either simultaneously in the parallel environment or independently in the sequential environment. An alternative approach is to solve them using block Krylov solvers for very small s. Note that c cannot be computed until iterations for (16) and (17) terminate but the stopping criterion (21) depends on c. Therefore, the computation of c and termination of iterations interacts. In implementations, we simply replace c by one in (21) and stop iterative solvers for the s systems (17) with the modified accuracy requirement. Because of this inaccuracy, (22) may fail to meet but $\|r\|/\|b\|$ should be at the level of ε generally. In later numerical experiments, we will find that $c = 1$ works very well and makes (22) hold for almost all the test problems and the right-hand sides of (22) are only a little bit bigger than ε in the very rare cases where (22) does not meet.

4 Numerical experiments

In this section, we test our approach and compare it with the algorithm that preconditions (1) by SPAI and PSAI(tol) directly and then uses Krylov iterative methods to solve the resulting preconditioned systems. We report the numerical experiments obtained by the Biconjugate Gradient Stablized (BiCGStab) with SPAI and PSAI(tol) preconditioning on (1) and (16), (17), respectively. Such combinations give rise to four algorithms. For brevity, we do not write BiCGStab explicitly and them Standard-SPAI, New-SPAI, and Standard-PSAI(tol) and New-PSAI(tol), abbreviated as S-SPAI, N-SPAI and S-PSAI(tol), N-PSAI(tol), respectively. We will first demonstrate the very considerable superiority of N-SPAI to S-SPAI and that of N-PSAI(tol) to S-PSAI(tol). Then we compare the performance of PSAI(tol) and SPAI.

We will illustrate that if the numbers of nonzero entries of preconditioners are required to be (almost) the same then PSAI(tol) is more effective for preconditioning than SPAI for both irregular and regular sparse linear systems. Under such restriction to the sparsity of preconditioners, we are able to compare the preconditioning effectiveness of PSAI(tol) and SPAI. We will show that PSAI(tol) captures the positions of large entries in A^{-1} and $\tilde{A}^{-1}$ more effectively than SPAI and thus generate better precondiotioners. For other Krylov solvers, such as BiCG, CGS and the restarted GMRES(20), we have obtained analogous numerical results, from which the same conclusions can be drawn. So it suffices to only report and evaluate the results obtained by BiCGStab.

Before testing our approach, we look into all the matrices in [7] and give illustrative information on where irregular sparse linear systems are from, how common they are in practice, how big s can be and how dense irregular columns. We divide matrices into their problem domains, sort them by percentage of matrices in that domain that are irregular. For example, there are 35 total semiconductor problems in the collection, of which 13 are regular and 22 irregular. Matrices labeled as "graphs" in the collection are excluded (many of those are irregular, but not all are a linear system). Here are details:

- 100%: 0 regular frequency-domain circuit simulation problem; 4 irregular frequency-domain circuit simulation problems
- 100%: 0 regular linear programming problem; 1 irregular linear programming problem
- 63%: 13 regular semiconductor device problems; 22 irregular semiconductor device problems
- 56%: 26 regular power network problems; 33 irregular power network problems
- 51%: 124 regular circuit simulation problems; 132 irregular circuit simulation problems
- 61%: 53 regular optimization problems; 82 irregular optimization problems
- 33%: 2 regular computer graphics/vision problems; 1 irregular computer graphics/vision problem
- 33%: 44 regular economic problems; 21 irregular economic problems
- 25%: 6 regular counter-example problems; 2 irregular counter-example problems
- 20%: 28 regular eigenvalue/model reduction problems; 7 irregular eigenvalue/model reduction problems
- 11%: 25 regular material problems; 3 irregular material problems
- 11%: 62 regular chemical process simulation problems; 8 irregular chemical process simulation problems
- 11%: 8 regular statistical/mathematical problems; 1 irregular statistical/mathematical problems
- 11%: 54 regular theoretical/quantum chemistry problems; 7 irregular theoretical/quantum chemistry problems
- 10%: 118 regular 2D/3D problems; 13 irregular 2D/3D problems
- 8%: 12 regular acoustics problems; 1 irregular acoustics problem

- 5%: 287 regular structural problems; 14 irregular structural problems
- 3%: 28 regular combinatorial problems; 1 irregular combinatorial problem
- 3%: 167 regular computational fluid dynamics problems; 5 irregular computational fluid dynamics problems
- 3%: 30 regular thermal problems; 1 irregular thermal problem
- 2%: 49 regular electromagnetic problems; 1 irregular electromagnetic problem
- 0%: 2 regular least squares problems; 0 irregular least squares problem
- 0%: 47 regular model reduction problems; 0 irregular model reduction problem
- 0%: 4 regular other problems; 0 irregular other problem
- 0%: 2 regular random 2D/3D problems; 0 irregular random 2D/3D problem
- 0%: 3 regular robotics problems; 0 irregular robotics problem

The above data illustrates that irregular sparse $Ax = b$'s are quite common and come from many applications. Below let us give more details on the information about the size of s and how dense irregular columns are. Figure 1 depicts what we are concerned with. In the top left plot, the x axis is the order of a square matrix. The y axis is the number s of dense columns. A circle is a matrix in the collection. The steep line is $s = n$, which is not actually achievable, and the flat line is $s = \sqrt{n}$. This figure only plots matrices with at least one dense column. In the top right figure, each dot is a square matrix. The x axis is the mean, i.e., the average number, of nonzero entries in each column, which equals $nnz(A)/n$. The y axis is the maximum number of nonzero entries in any column divided by the mean for that matrix. The line parallel to the x axis is $y = 10$. Matrices have at least one dense column if they reside above the line $y = 10$. For each x, the bigger y, the denser the irregular column. The bottom 2 figures are the same, but with social networks and other graphs excluded. [2]

These figures further demonstate that there are very dense columns for many matrices, irregular sparse matrices are common and s can be quite big.

We test our approach on some of the above irregular sparse linear systems. A brief description is presented in Table 1. The right-hand side b of $Ax = b$ was formed by taking the solution $x = (1, 1, \ldots, 1)^T$. Taking the initial approximate solution to be zero for each problem and $\varepsilon = 10^{-8}$ in stopping criterion (22), we have found that BiCGStab without preconditioning did not converge for all the test problems within 500 iterations. We note that cbuckle is symmetric. So it should be better to design preconditioners to maintain the symmetry of preconditioned matrices, which can be achieved by using factorized or splitting preconditioners, so that Krylov solvers for symmetric problems can be applied. In our experiments, however, we used cbuckle in PSAI(tol) purely for test propose and treated it as a general matrix.

We conducted the numerical experiments on an Intel (R) Core (TM)2 Quad CPU E8400 @ 3.00GHz with main memory 2 GB under the Linux operating system. The computations were done using MATLAB 7.8.0 with the machine precision $\epsilon_{\text{mach}} = 2.22 \times 10^{-16}$, and SPAI preconditioners were constructed by the SPAI 3.2 package [2] of Barnard, Bröker, Grote and Hagemann, which is written in C/MPI. We used both SPAI and PSAI(tol) as right-preconditioning and took the initial patterns $\mathcal{S}_k^{(0)} = \{k\},\ k = 1, 2, \ldots, n$. We took $c = 1$ in

[2]The figures and the previous data analysis are due to Professor Davis, and we thank him very much.

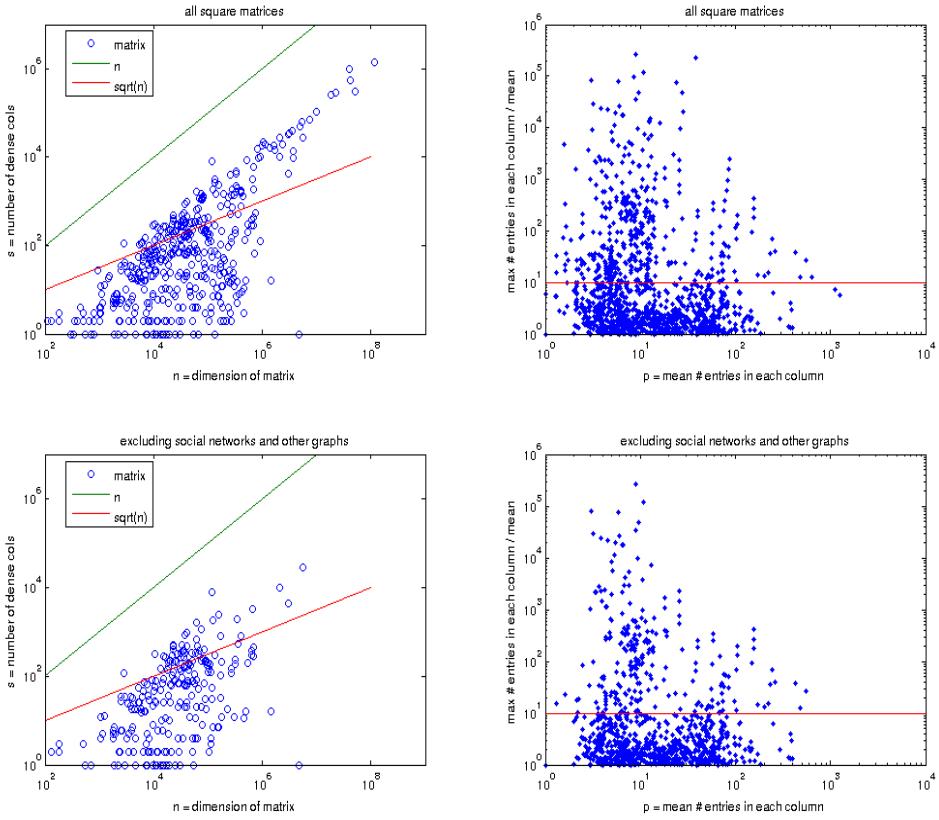

Figure 1: Illustrations of irregular matrices in [7].

(21) and stopped iterations when (20) and (21) were satisfied with $\varepsilon = 10^{-8}$ or 500 iterations were reached. The initial approximate solution for each problem was zero vector. With $\tilde{x}$ defined by (19), we computed the actual relative residual norm

$$rr = \frac{\|b - A\tilde{x}\|}{\|b\|} \tag{24}$$

and compared it with the required accuracy $\varepsilon = 10^{-8}$.

Before performing our algorithms, we carried out row Dulmage–Mendelsohn permutations [8, 16] on the matrices whose diagonals have zero elements, so as to make their diagonals nonzero whenever necessary. The related MATLAB commands are $j = \texttt{dmperm}(A)$ and $A = A(j,:)$. We applied `dmperm` to rajat04, rajat12, tols4000 and ASIC_100k. The threshold for defining an irregular column is $10p$. For all the matrices listed in Table 1, $\tilde{A}$ is constructed by retaining the diagonal entry and $p-1$ nonzero entries nearest to the diagonal and dropping the others in each of the irregular columns of A, and U is composed of the dropped nonzero entries. Table 2 shows some useful information about each A and $\tilde{A}$, including the order n, the number s of irregular columns in A, the average number p_a of nonzero entries per column of A, the number p_d of nonzero entries in the densest column of A_{dc}, the numbers $nnz(\cdot)$ of nonzero entries of A and $\tilde{A}$ and the 2-norm condition numbers or their 1-norm estimates of A and $\tilde{A}$.

We have some informative observations from Table 2. As is seen, except cbuckle and

Table 1: The description of test matrices, where $\kappa(A) = \|A\|\|A^{-1}\|$, 'sym' denotes the numeric value symmetry and we used the MATLAB function condest to estimate the 1-norm condition numbers of the latter seven larger matrices.

Matrix	n	nnz	$\kappa(A)$	sym	Description
fs_541_3	541	4282	2.83×10^{11}	No	2D/3D problem
fs_541_4	541	4273	1.17×10^{10}	No	2D/3D problem
rajat04	1041	8725	1.64×10^{8}	No	circuit simulation problem
rajat12	1879	12818	6.91×10^{5}	No	circuit simulation problem
tols4000	4000	8784	2.36×10^{7}	No	computational fluid dynamics problem
cbuckle	13681	676515	3.30×10^{7}	Yes	structural problem
ASIC_100k	99340	940621	1.46×10^{11}	No	circuit simulation problem
dc1	116835	766396	1.01×10^{10}	No	circuit simulation problem
dc2	116835	766396	8.86×10^{9}	No	circuit simulation problem
dc3	116835	766396	1.16×10^{10}	No	circuit simulation problem
trans4	116835	749800	3.30×10^{9}	No	circuit simulation problem
trans5	116835	749800	2.32×10^{9}	No	circuit simulation problem

tols4000, all the other test matrices have some almost fully dense columns. The matrix cbuckle and tols4000 are not so bad, but they indeed have one and 18 irregular columns under our definition. Precisely, except cbuckle and tols, raja04 and raja12 have some columns that have more than $n/2$ nonzero entries, and all the other matrices have some fully dense columns. Table 2 shows that each A has a few irregular columns. The biggest two percentages 0.45% and 0.12% , which are equal to s/n, are reached for tols4000 and ASIC_100k, where $s = 22$ and 122, respectively. It is remarkable that the eight $\tilde{A}$'s are considerably better conditioned than the corresponding A's, where the condition numbers of $\tilde{A}$'s are reduced by roughly one to five orders, compared with those of A's. For the other three matrices rajat04, rajat12 and cbuckle, $\tilde{A}$'s and A's have very near condition numbers.

We next investigate the patterns of large entries in the "exact" inverses of an irregular sparse matrix and the regular sparse matrix induced from it. We only take rajat04 as an example. After performing row Dulmage–Mendelsohn permutation on it, we use the MATLAB function `inv` to compute A^{-1} and $\tilde{A}^{-1}$ and then drop their nonzero entries whose magnitudes fall below 10^{-3}. We depict the patterns of sparsified A^{-1} and $\tilde{A}^{-1}$ as (A) and (B) in Figure 2, respectively. It is clear that good approximate inverses of A and $\tilde{A}$ are sparse but there are several dense columns in (A), which means that an effective sparse approximate inverse of A is irregular. On the other hand, the situation is improved substantially in (B), from which it is seen that a good sparse approximate inverse of $\tilde{A}$ is regular sparse and it is sparser than the matrix in (A). These results typically demonstrate that good sparse approximate inverses of an irregular sparse matrix are also irregular but in contrast a regular sparse matrix has regular sparse approximate inverses. We have checked several other test matrices and have had the same findings. Perceptually, all of these have strongly supported our empirical assertions that the good sparse approximate inverses of an irregular or regular sparse matrix are very possibly irregular or regular sparse too.

Here and hereafter, in all the tables, *pt* and *st* are the total CPU timings (in seconds) of constructing M and solving the preconditioned linear systems by BiCGStab, respectively, $spar = nnz(M)/nnz(A)$ or $nnz(M)/nnz(\tilde{A})$ denotes the sparsity of M relative to A or $\tilde{A}$, and *iter* stands for the iteration number that BiCGStab used for (1) and the maximum of iteration numbers that BiCGStab used for the $s+1$ systems (16) and (17), respectively. The

Table 2: Some information about A and $\tilde{A}$, where we used the MATLAB function condest to estimate the 1-norm condition numbers of the latter seven larger matrices.

	n	s	p_a	p_d	$nnz(A)$	$nnz(\tilde{A})$	$\kappa(A)$	$\kappa(\tilde{A})$
fs_541_3	541	1	7	538	4282	3745	2.83×10^{11}	7.86×10^{6}
fs_541_4	541	1	7	535	4273	3739	1.17×10^{10}	1.64×10^{6}
rajat04	1041	4	8	642	8725	7306	1.64×10^{8}	1.16×10^{8}
rajat12	1879	7	6	1195	12818	9876	6.91×10^{5}	6.53×10^{5}
tols4000	4000	18	2	22	8784	8424	2.36×10^{7}	2.36×10^{7}
cbuckle	13681	1	49	600	676515	675916	3.30×10^{7}	8.06×10^{7}
ASIC_100k	99340	122	9	92258	940621	742736	1.46×10^{11}	9.28×10^{9}
dc1	116835	55	6	114174	766396	595757	1.01×10^{10}	2.17×10^{8}
dc2	116835	55	6	114174	766396	595757	8.86×10^{9}	5.81×10^{7}
dc3	116835	55	6	114174	766396	595757	1.16×10^{10}	1.52×10^{8}
trans4	116835	55	6	114174	749800	587459	3.30×10^{9}	3.46×10^{8}
trans5	116835	55	6	114174	749800	587459	2.32×10^{9}	6.54×10^{7}

actual residual norm rr defined by (24) is $a \cdot \varepsilon$, and we only list the multiple a in the tables. So $a < 1$ indicates that BiCGStab converged with the prescribed accuracy ε. The size of a reflects wether taking $c = 1$ in (21) is reliable or not.

4.1 Numerical results obtained by SPAI

We take $\delta = 0.4$ as the stopping criterion for SPAI. Since effective sparsity patterns of A^{-1} and $\tilde{A}^{-1}$ are generally unknown in advance, some key parameters, especially the number of most profitable indices per loop, involved in Procedure * can only be chosen empirically in order to control the sparsity of M and simultaneously to make M achieve the desired accuracy δ as far as possible. In the experiments of this subsection, we fix the number of most profitable indices to be five per loop, which is also the default value in the code. The maximum number of loop steps $l_{\max}$ is set as twenty, bigger than the default value five in the code. The SPAI terminated whenever $\|Am_k - e_k\| < \delta,\ k = 1, 2, \ldots, n$ or the the maximum loop steps $l_{\max}$ was attained. It is run similarly for $\tilde{A}$. With the given parameters and the initial pattern $\mathcal{S}_k^{(0)} = \{k\},\ k = 1, 2, \ldots, n$, the number of nonzero entries per column in the final M is bounded by $1 + 5 \times 19 = 96$. So if good preconditioners are irregular sparse, then SPAI may not be effective for preconditioning. We mention that we have omitted the symmetric matrix cbuckle since the matrix input in the SPAI 3.2 package is only supported by "real general" Matrix Market coordinate format. Table 3 lists the results.

We make some comments on the results. First, we look at the efficiency of constructing M's by S-SPAI and N-SPAI. The notation '$*$' for the last six larger matrices indicates that S-SPAI could not compute M's within 100 hours because of the irregular sparsity of A's. Since each of these six matrices has fully dense irregular columns, the unaffordable time consumption results from the large cardinal numbers of $\tilde{\mathcal{J}}$'s and $\mathcal{L}$'s in Section 2.1. More precisely, for a fully dense irregular column of A, we have to compute almost n numbers μ_j's in (7), sort almost n indices in $\tilde{\mathcal{L}}$ and select five most profitable indices among them at each loop. Carrying out these tasks is very time consuming. In contrast, N-SPAI did a very good job due to the regular sparsity of $\tilde{A}$'s. For the other matrices and given the parameters, the two M's obtained by S-SPAI and N-SPAI have very similar sparsity, but it is seen from pt's that N-SPAI can be considerably more efficient to compute M's than S-SPAI, and the former may make great improvements over the latter and can be several times faster, e.g., six

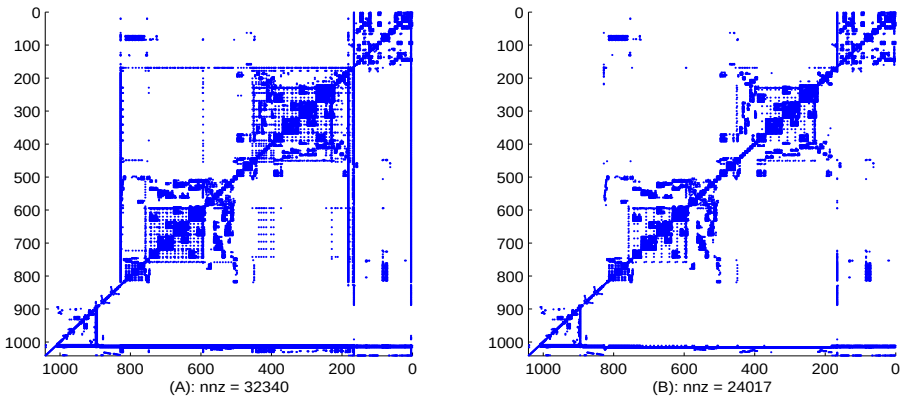

Figure 2: rajat04: (A) and (B) are the patterns of sparsified A^{-1} and $\tilde{A}^{-1}$, respectively.

and seven times faster for raja04 and raja12, respectively. For the last six matrices, N-SPAI computed all the M's within no more than half an hour to two hours, drastic improvements over S-SPAI!

Second, we investigate the effectiveness of M's for preconditioning. To be more illuminating, we recorded the number n_c of columns in M which do not satisfy the accuracy $\delta = 0.4$ for each matrix in S-SPAI or N-SPAI. We have observed that for the first four matrices, n_c's produced by N-SPAI are always smaller than those by S-SPAI. To some extent, this illustrates that it is more difficult for SPAI to compute an effective preconditioner when A is irregular sparse. Indeed, as indicated by *iter*'s, it is seen that for the first four matrices N-SPAI is more effective than S-SPAI. This confirms our claim that SPAI may be less effective for preconditioning irregular sparse linear systems.

Third, we have found from a's that most of all the actual relative residual norms rr's in (24) dropped below ε. The case that $a > 1$ happened only for ASIC_100k, but the actual relative residual norm $rr = 1.37 \times 10^{-8}$ is already very near to the required accuracy $\varepsilon = 10^{-8}$, indicating that the algorithm converged essentially. This means that taking $c = 1$ in (21) worked very well in practice. Note that *iter* for N-SPAI simply denotes the maximum of the iterations that BiCGStab used for $s+1$ preconditioned linear systems. Therefore, it indicates the worst case but cannot tell us the convergence behavior of BiCGStab for each of the $s+1$ systems. Actually, for dc1, we have observed that only two of the s systems (17) needed considerably more iterations than the others. The reason is that the two $\|b\|/\|u_j\|$'s involved in the right-hand side of (21) are at least one order smaller than those of the remaining systems, so that more iterations were needed.

Finally, we compare the overall performance of N-SPAI and S-SPAI. Naturally, for the last six larger matrices, S-SPAI failed to compute M's within 100 hours while N-SPAI was very efficient to do the same job and exhibited a very substantial superiority. For the other problems, N-SPAI is two to eight times as fast as S-SPAI to compute M's. We see that the

Table 3: Numerical results obtained by SPAI

	S-SPAI						N-SPAI						
	pt	*st*	*spar*	*iter*	*a*	n_c	*pt*	*st*	*spar*	*iter*	*a*	n_c	*s*
fs_541_3	0.33	0.01	1.34	21	0.40	1	0.17	0.01	1.52	8	0.62	0	1
fs_541_4	0.14	0.01	0.81	14	0.01	1	0.04	0.01	0.91	6	0.68	0	1
rajat04	1.17	0.01	0.36	31	0.20	6	0.16	0.02	0.40	17	0.57	2	4
rajat12	3.19	0.01	0.86	48	0.31	3	0.39	0.08	1.08	39	0.80	0	7
tols4000	0.10	0.02	1.01	3	0.44	6	0.05	0.02	1.02	4	0.48	6	18
ASIC_100k	*	*	*	*	*	*	1619	13.52	0.66	10	1.37	12	122
dc1	*	*	*	*	*	*	8216	44.07	1.71	351	0.61	1	55
dc2	*	*	*	*	*	*	6210	41.75	1.62	76	0.75	0	55
dc3	*	*	*	*	*	*	5243	47.05	1.63	98	0.91	9	55
trans4	*	*	*	*	*	*	3798	9.13	1.64	39	0.49	0	55
trans5	*	*	*	*	*	*	3546	20.79	1.57	84	0.65	0	55

construction of M's by N-SPAI dominates the total cost of our approach and the efficiency of N-SPAI compensates for the price of solving $s+1$ linear systems. As a result, as far as the overall efficiency is concerned, our approach has exhibited a great superiority to SPAI applied to precondition (1) directly.

4.2 Numerical results obtained by PSAI(tol)

We look into the performance of N-PSAI(tol) and S-PSAI(tol) and show that the former is much better than the latter. In PSAI(tol), we always take $\delta = 0.4$ and $l_{\max} = 10$ to control the sparsity and quality of M for both S-PSAI(tol) and N-PSAI(tol). PSAI(tol) terminates when $\|Am_k - e_k\| < \delta$ or the loop steps $l > l_{\max}$. We use (10) as the dropping tolerances. We test all the matrices in Table 1 and report the results in Table 4, where l_m is the actual maximum loop steps used for computing all the columns $m_k,\ k = 1, 2, \ldots, n$ of M.

Table 4: Numerical results obtained by PSAI(tol)

	S-PSAI(tol)						N-PSAI(tol)						
	pt	*st*	*spar*	*iter*	*a*	l_m	*pt*	*st*	*spar*	*iter*	*a*	l_m	*s*
fs_541_3	0.36	0.01	1.55	6	0.31	3	0.29	0.01	1.63	6	0.30	4	1
fs_541_4	0.30	0.01	1.35	5	0.22	3	0.25	0.01	1.38	5	0.42	3	1
rajat04	1.27	0.01	0.72	11	0.79	4	0.28	0.02	0.51	11	0.58	4	4
rajat12	3.48	0.01	2.22	32	0.74	2	1.40	0.10	1.92	44	0.56	2	7
cbuckle	3841	1.62	3.41	85	0.21	5	2322	2.12	2.76	71	0.55	5	1
tols4000	0.94	0.01	0.97	2	0.11	1	0.79	0.01	1.00	2	0.23	2	18
ASIC_100k	-	-	-	-	-	-	1429	10.05	0.81	6	0.78	2	122
dc1	-	-	-	-	-	-	3203	31.36	1.54	201	0.81	3	55
dc2	-	-	-	-	-	-	2975	27.33	1.62	59	0.58	3	55
dc3	-	-	-	-	-	-	2966	24.10	1.63	56	0.61	8	55
trans4	-	-	-	-	-	-	3073	5.61	1.79	31	0.44	4	55
trans5	-	-	-	-	-	-	2856	9.86	1.61	58	1.29	4	55

The notation '-' for the last six larger matrices indicates that our computer was out of memory when constructing each M. The cause is that each A of them has some fully dense irregular columns, which result in some large LS problems (4). So, PSAI(tol) may encounter

an essential difficulty when A has some very dense columns. For all the other smaller matrices, S-PSAI(tol) can generate M's with the desired accuracy $\delta = 0.4$. Furthermore, we have seen that S-PSAI(tol) always used nearly the same l_m as N-PSAI(tol) for each A but it is considerably more time consuming than N-PSAI(tol). So, PSAI(tol) can capture effective sparse patterns of A^{-1} and $\tilde{A}^{-1}$ for (almost) the same small loop steps l and determine large entries of them reliably. This is a distinctive feature that S-SPAI and N-SPAI do not share, where SPAI may be ineffective for preconditioning when A is irregular sparse and the situation is improved when A is regular sparse.

We highlight more on the effectiveness for preconditioning. As we have seen from Table 4, S-PSAI(tol) and N-PSAI(tol) provide comparably effective preconditioners for the first five matrices since BiCGStab used comparable iterations $iter$'s to achieve the convergence. This is expected, as M's obtained by S-PSAI(tol) have similar accuracy to those obtained by N-PSAI(tol) correspondingly. But we should be aware that, for the same accuracy δ, S-PSAI(tol) can generate an M that is considerably denser than that obtained by N-PSAI(tol); see, e.g., cbuckle. This shows that good sparse approximate inverses of an irregular sparse matrix A is denser that those of $\tilde{A}$ but PSAI(tol) can capture effective sparsity patterns of A^{-1}. This is an advantage of PSAI(tol) over SPAI for A irregular sparse.

We comment that taking $c = 1$ works very well and makes almost all the actual relative residual norms defined by (24) drop ε. The only exception is for trans5, for which $a = 1.29$. But such a indicates that the actual relative residual norm for this problem is very near to ε, so we can well accept the approximate solution as essentially converged.

Finally, we are concerned with the overall performance of the algorithms used. The table clearly shows that the performance of N-PSAI(tol) is very superior to S-PSAI(tol), in terms of the total computational time that is the sum of pt and st. Since S-PSAI(tol) and N-PSAI(tol) provide equally effective preconditioners with comparable sparsity for each A and $\tilde{A}$, it is natural that the solving time st of N-PSAI(tol) is more than that of S-PSAI(tol) by noting that N-PSAI(tol) solves the $s + 1$ linear systems. Even so, however, the time st of solving $s + 1$ systems is negligible to pt for each problem.

4.3 Effectiveness comparison of S-SPAI and S-PSAI(tol) and that of N-SPAI and N-PSAI(tol)

We attempt to give some comparison of the effectiveness of S-SPAI and S-PSAI(tol) and that of N-SPAI and N-PSAI(tol). To do so, we take the sparsity as a reference. As we have known, all M's in Table 4 satisfy the desired accuracy $\delta = 0.4$. Now, for each A, we adjust the number of most profitable indices per loop and the maximum loop steps $l_{\max}$, so that N-SPAI generates an M whose sparsity is approximately equal to that of the corresponding preconditioner obtained by N-PSAI(tol) in Table 4. We then perform S-SPAI with the same parameters to compute a sparse approximate inverse of A's. We aim to show that, given the (almost) same sparsity of preconditioners, N-PSAI(tol) may capture sparsity patterns more effectively and produces better preconditioners than N-SPAI does. The same conclusions hold for S-PSAI(tol) and S-SPAI.

To make each M by N-SPAI be (almost) equally sparse as that obtained by N-PSAI(tol) for each $\tilde{A}$, we take the parameters in the SPAI 3.2 code as

fs_541_3 '-mn 6 -ns 15'
fs_541_4 '-mn 6 -ns 20'
rajat04 '-mn 8 -ns 20'
rajat12 '-mn 12 -ns 10'
tols4000 '-mn 5 -ns 20'

Table 5: Numerical results obtained by SPAI

	S-SPAI					N-SPAI					
	pt	*st*	*spar*	*iter*	*a*	*pt*	*st*	*spar*	*iter*	*a*	*s*
fs_541_3	0.32	0.01	1.65	48	0.72	0.20	0.01	1.64	20	0.40	1
fs_541_4	0.15	0.01	0.96	15	0.11	0.03	0.01	0.94	7	0.34	1
rajat04	1.32	0.01	0.56	20	0.99	0.16	0.02	0.50	12	0.77	4
rajat12	2.71	0.02	2.05	42	0.39	0.42	0.09	1.99	29	0.67	7
tols4000	0.10	0.02	1.01	3	0.44	0.05	0.02	1.02	4	0.48	18
ASIC_100k	78h	10.16	0.65	500	1560	1807	12.61	0.76	10	0.92	122
dc1	*	*	*	*	*	7275	88.25	1.66	500	0.73	55
dc2	*	*	*	*	*	6208	54.33	1.62	90	0.57	55
dc3	*	*	*	*	*	5198	70.63	1.63	422	1.36	55
trans4	*	*	*	*	*	3571	9.23	1.82	40	0.47	55
trans5	*	*	*	*	*	3033	19.52	1.71	81	0.82	55

ASIC_100k '-mn 6 -ns 2'
dc1 '-mn 5 -ns 7'
dc2 '-mn 5 -ns 7'
dc3 '-mn 5 -ns7'
trans4 '-mn 6 -ns 15'
trans5 '-mn 6 -ns 7'

where $ns = l_{\max}$ in our notation with '-ns j' denoting $ns = j$ and mn is the number of most profitable indices per loop with '-mn j' denoting $mn = j$. Table 5 reports the results.

Based on Tables 4–5, we can compare the effectiveness of S-SPAI and S-PSAI(tol) and that of N-SPAI and N-PSAI(tol), respectively.

Obviously, N-PSAI(tol) improves on N-SPAI for all the test matrices but $\tilde{A}$ resulting from rajat12. The results on the last six larger matrices are more illustrative, where PSAI(tol) exhibited the considerable superiority to SPAI for regular sparse linear systems. Particularly, for dc1, when N-SPAI is applied, BiCGStab used just 500 iterations to achieve the stopping criterion (21) with $c = 1$, while N-PSAI(tol) only used 200 iterations; for dc3, N-PSAI(tol) produced a much more effective preconditioner than N-SPAI, and BiCGStab preconditioned by N-PSAI(tol) is seven times faster than that by N-SPAI. The cause is just that given almost the same sparsity, PSAI(tol) better captures effective sparsity patterns, i.e., the positions of large entries of A^{-1} and $\tilde{A}^{-1}$ than SPAI does. This confirms our arguments in the introduction and [15].

For the direct application to the original irregular sparse (1), S-PSAI(tol) shows greater improvements over S-SPAI. For ASIC_100k, S-SPAI consumed 78 hours to construct a sparse approximate inverse M. But M is of poor quality and BiCGStab failed to converge after 500 iterations with the actual relative residual norm $rr = 1.56 \times 10^{-3}$. The reason should be that some columns of M are too sparse to capture enough positions of the large entries in the corresponding columns of A^{-1}. For the first four matrices, we see from Tables 4–5 that two M's for each A have very comparable sparsity but the results clearly illustrate that, with a comparable sparsity, S-PSAI(tol) is considerably more effective for preconditioning than S-SPAI for the four matrices. It is eight times, three times, twice and nearly one and a half times as fast as S-SPAI for the four problems, respectively, as the corresponding *iter*'s indicate. So S-PSAI(tol) results in a more substantial acceleration of BiCGStab than S-

SPAI. This demonstrates that PSAI(tol) captures sparsity patterns of A^{-1} more effectively than SPAI for A irregular sparse and compute the large entries of A^{-1} reliably, thus leading to more effective preconditioners than SPAI when A is irregular sparse.

Summarizing the above, we conclude that PSAI(tol) itself is effective for preconditioning no matter whether a matrix is regular sparse or not while SPAI appears to be more suitable for regular sparse matrices and may be ineffective when A is irregular sparse. Even for regular sparse linear systems, PSAI(tol) can outperform SPAI considerably for preconditioning. Taking the construction cost of preconditioners by SPAI and PSAI(tol) into account, to make them computationally practical, we should apply them to regular sparse linear systems. Therefore, for an irregular sparse linear system, a good means is to transform it into some regular sparse problems, so that SPAI and PSAI(tol) are relatively efficient to compute effective sparse approximate inverses.

As a last note, we make some comments on the efficiency of SPAI and PSAI(tol). Since they are different procedures that are derived from different principles and have different features, the computational complexity of each of them is quite involved, and the efficiency depends on several factors including pattern of A itself. It appears very hard, if not impossible, to analyze and compare their costs. Hence we cannot draw any definitive conclusion on the efficiency. However, it is seen from Tables 3–5 that for given reasonably parameters N-PSAI(tol) was at least comparable to N-SPAI in efficiency and consumed less CPU time to construct M's. As far as the computational efficiency is concerned, there must be cases where one procedure wins the other and vice versa. Here we should point out that our PSAI(tol) is written in the MATLAB language in single processor while SPAI 3.2 is more or less optimized and better programmed in C/MPI designed for parallel computers. So, the SPAI 3.2 code itself is better programming than PSAI(tol) written in MATLAB even in the sequential environment.

A parallel PSAI(tol) code in C/MPI or Fortran is involved and will be left as our future work. We will expect that the performance of PSAI(tol) is improved greatly.

Finally, we point out that our numerical results are for the threshold $10p$, by which we define irregular columns and regard a column as irregular if the number of its nonzero entries exceeds $10p$. This threshold is empirical and can be flexible. Actually, we have found that our approach is quite insensitive to it. A threshold between $8p \sim 15p$ works effectively too, that is, the resulting $\tilde{A}$'s have similar features as before, and we can obtain similar results by using N-SPAI and N-PSAI(tol) and draw similar conclusions on them.

5 Conclusions

The SPAI and PSAI(tol) algorithms are quite effective in accelerating Krylov subspace solvers for a wide range of real world problems. However, the situation is rather disappointing for quite common irregular sparse linear systems. In this case, none of SPAI and PSAI(tol) works well generally due to the very high cost and/or possibly excessive storage requirement of constructing preconditioners and/or the ineffectiveness of preconditioners. However, for a regular sparse matrix, we have shown that it is possible to use SPAI and PSAI(tol) to construct effective preconditioners efficiently. Motivated by this crucial feature and exploiting the Sherman–Morrison–Woodbury formula, we have transformed the original irregular sparse linear system into some regular sparse ones, for which SPAI and PSAI(tol) are practical. We have considered some practical issues and derived stopping criteria for iterative solutions of new systems, so that the approximate solution of the original problem achieves the user prescribed accuracy. In such a way, we have extended the applicability of SPAI and PSAI(tol) very significantly. Numerical experiments on a number of real world problems with A irreg-

ular sparse have demonstrated that our approach works very well and has drastic effects on the overall performance of SPAI and PSAI(tol) preconditioning, compared with SPAI and PSAI(tol) applied to the original (1) directly.

Our approach may be applicable to factorized sparse approximate inverse preconditioning procedures [3]. Although reorderings of A may help such kind of procedures to reduce fill-ins, enhance robustness and improve numerical stability when constructing factorized SAI preconditioners, they are not always so and in fact even may make things worse for A irregular sparse [6]. We believe that our approach may be combined with reorderings to construct effective factorized SAI preconditioners for regular sparse linear systems resulting from the irregular sparse one.

We have demonstrated that our approach is more practical and for irregular sparse linear systems. Even for A very large with a large number of irregular sparse columns, our approach work effectively as the cost of constructing preconditioners overwhelms the cost of solving linear systems themselves provided that preconditioners are effective. One might notice that the right-hand side of (21) is inversely proportional to $\sqrt{s}$ and smaller than the required accuracy ε. Nevertheless, the higher accuracy requirements on the s linear systems do not cause any difficulty since they decrease quite slowly with s and are no more than two or three orders smaller than ε even for s ranging from $10^4 \sim 10^6$ (note such s means that A is very large or huge).

References

[1] S.T. Barnard, L.M. Bernardo and H.D. Simon, An MPI implementation of the SPAI preconditioner on the T3E, *Intern. High Perform. Comput. Appl.*, 13 (1999): 107–123.

[2] S.T. Barnard, O. Broker, M.J. Grote and M. Hagemann, SPAI 3.2 package, http://www.computational.unibas.ch/software/spai, 2006.

[3] M. Benzi, Preconditioning techniques for large linear systems, *J. Comput. Phys.*, 182 (2002): 418–477.

[4] M. Benzi and M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM J. Sci. Comput.*, 21 (1998): 968–994.

[5] M. Benzi and M. Tůma, Orderings for factorized sparse approximate inverse preconditioners, *SIAM J. Sci. Comput.*, 21 (2000): 1851–1868.

[6] M. Benzi and M. Tůma, A robust preconditioner with low memory requirements for large sparse least squares problems, *SIAM J. Sci. Comput.*, 25 (2003): 499–512.

[7] T.A. Davis and Y. Hu, The University of Florida sparse matrix collection, *ACM Trans on Math. Soft. (TOMS)*, 38 (2011): 1–25.

[8] I.S. Duff, On algorithms for obtaining a maximum transversal, *ACM Trans. Math. Soft.*, 7 (1981): 315–330.

[9] R.W. Freund, G.H. Golub and N. Nachtigal, Iterative solution of linear systems, *Acta Numerica*, 1 (1992): 57–100.

[10] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, The John-Hopkins University Press, Baltimore, MD, 1996.

[11] M.J. Grote and T. Huckle, Parallel preconditioning with sparse approximate inverses, *SIAM J. Sci. Comput.*, 18 (1997): 838–853.

[12] W.W. Hager, Updating the inverse of a matrix, *SIAM Review*, 31 (1989): 221–239.

[13] T. Huckle, Approximate sparsity patterns for the inverse of a matrix and preconditioning, *Appl. Numer. Math.*, 30 (1999): 291–303.

[14] Z. Jia and Q. Zhang, Robust dropping criteria for F-norm minimization based sparse approximate inverse preconditioning, arXiv: math/1203.2325v2, 2012.

[15] Z. Jia and B. Zhu, A power sparse approximate inverse preconditioning procedure for large sparse linear systems, *Numer. Linear Algebra Appl.*, 16 (2009): 259–299.

[16] A. Pothen and C.-J. Fan, Computing the block triangular form of a sparse matrix, *ACM Trans. Math. Soft.*, 16 (1990): 303–324.

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2003.

[18] G.W. Stewart, *Matrix Algorithms Volume I: Basic Decompositions*, SIAM, Philadelphia, PA, 1998.